A3

PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 :		(11) International Publication Number:	WO 97/24667
G06F 11/14	A1	(43) International Publication Date:	10 July 1997 (10.07.97)

(21) International Application Number: PCT/US96/19927

(22) International Filing Date: 27 December 1996 (27.12.96)

(30) Priority Data:
08/579,810 28 December 1995 (28.12.95) US

(71) Applicant: IPL SYSTEMS, INC. [US/US]; 124 Acton Street, Maynard, MA 01754 (US).

(72) Inventors: CAVALLO, Joseph, S.; 87 Waltham Street, Maynard, MA 01754 (US). IPPOLITO, Stephen, J.; 594 Old Marlboro Road, Concord, MA 01742 (US). SCHARLAND, Michael, J.; 49 Long Avenue, Framingham, MA 01701 (US).

(74) Agent: KUDIRKA, Paul, E.; Bookstein & Kudirka, P.C., One Beacon Street, Boston, MA 02108 (US).

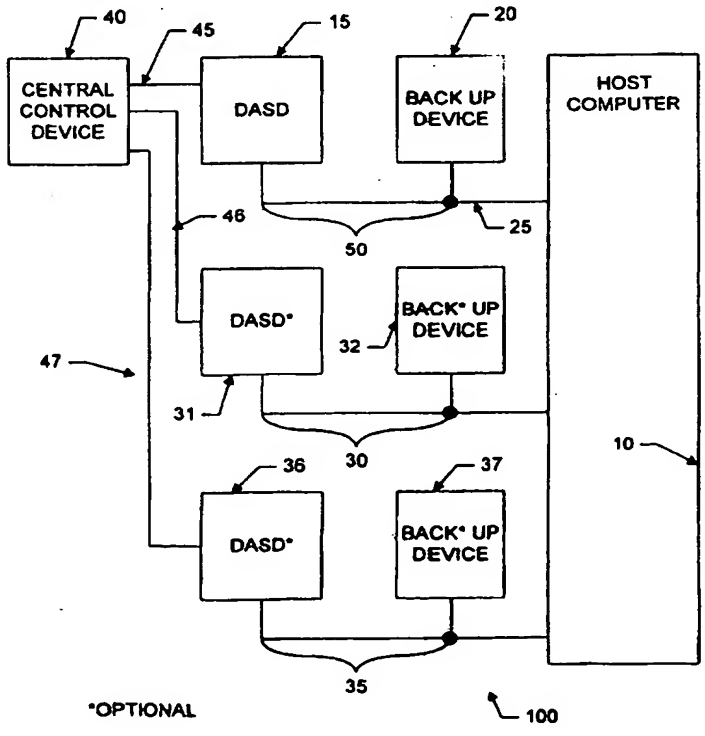
(81) Designated States: CA, JP, European patent (AT, BE, CH, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE).

Published
With international search report.
Before the expiration of the time limit for amending the claims and to be republished in the event of the receipt of amendments.

(54) Title: DASD STORAGE BACK-UP USING OUT-OF-SEQUENCE WRITES

(57) Abstract

A back-up system. The back-up system sequences through DASD locations and stores the DASD data. The DASD records which DASD locations have been backed-up. The DASD concurrently receives and performs CPU activity including write transactions. If a write transaction is received that targets a DASD location that has not yet been backed up, the DASD backs up the data block at that target location, out of sequence. The DASD then resumes with sequencing through the DASD locations and concurrently performing normal CPU activity. The invention may be realized in a variety of system configurations. The inventive logic is preferably realized in the DASD, but may also be realized in the CPU.



FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AM	Armenia	**GB**	United Kingdom	**MW**	Malawi
AT	Austria	**GE**	Georgia	**MX**	Mexico
AU	Australia	**GN**	Guinea	**NE**	Niger
BB	Barbados	**GR**	Greece	**NL**	Netherlands
BE	Belgium	**HU**	Hungary	**NO**	Norway
BF	Burkina Faso	**IE**	Ireland	**NZ**	New Zealand
BG	Bulgaria	**IT**	Italy	**PL**	Poland
BJ	Benin	**JP**	Japan	**PT**	Portugal
BR	Brazil	**KE**	Kenya	**RO**	Romania
BY	Belarus	**KG**	Kyrgystan	**RU**	Russian Federation
CA	Canada	**KP**	Democratic People's Republic of Korea	**SD**	Sudan
CF	Central African Republic			**SE**	Sweden
CG	Congo	**KR**	Republic of Korea	**SG**	Singapore
CH	Switzerland	**KZ**	Kazakhstan	**SI**	Slovenia
CI	Côte d'Ivoire	**LI**	Liechtenstein	**SK**	Slovakia
CM	Cameroon	**LK**	Sri Lanka	**SN**	Senegal
CN	China	**LR**	Liberia	**SZ**	Swaziland
CS	Czechoslovakia	**LT**	Lithuania	**TD**	Chad
CZ	Czech Republic	**LU**	Luxembourg	**TG**	Togo
DE	Germany	**LV**	Latvia	**TJ**	Tajikistan
DK	Denmark	**MC**	Monaco	**TT**	Trinidad and Tobago
EE	Estonia	**MD**	Republic of Moldova	**UA**	Ukraine
ES	Spain	**MG**	Madagascar	**UG**	Uganda
FI	Finland	**ML**	Mali	**US**	United States of America
FR	France	**MN**	Mongolia	**UZ**	Uzbekistan
GA	Gabon	**MR**	Mauritania	**VN**	Viet Nam

DASD STORAGE BACK-UP USING OUT-OF-SEQUENCE WRITES

Field of the Invention

The instant invention generally relates to storage technology and, more particularly, to improvements in backing up the state of a computing system.

Background

Modern computing systems use direct access storage devices (DASDs) to non-volatilely store information provided by a central processing unit (CPU). Known DASDs, such as the IPL ESP 7037, sold by IPL Systems, Inc., of Maynard Massachusetts, typically include a controller board and several storage devices, such as disks sold by Seagate Technology and Digital Equipment Corp. The controller board typically includes a microprocessor and a memory holding firmware, which is executed by the microprocessor to perform the various DASD functions. The controller firmware, for example, includes write and read logic, which maps an address included with a read or write command to a corresponding location on one of the storage devices and performs the requested command. In effect, a DASD appears as one large accessible storage area to the host CPU, rather than as a conglomeration of the various storage devices associated with and controlled by the DASD.

Host CPUs conventionally organize and use data as files, arranged as blocks of data. Software executing on the CPU operates on stored data by requesting file-level operations, e.g., "open file *x*; read a record *y* from file *x*; etc." An operating system, or other system-level software, translate this file-level operation into a disk-level operations. For example, "read record *y* from file *x*" may be translated as "go to disk *m* and read block *n* from sector *l*," in effect, specifying a disk and a location on that disk.

DASDs handle "logical disks." That is, a DASD may actually have *p* physical storage devices under its control, but this fact is hidden from the perspective of the CPU. Instead, the CPU may perceive the DASD as an arrangement of *q* disks; the *q* disks are the logical disks. Any requests to one of the *q* logical disks is handled by the

DASD. Among other things, the DASD will map the request to one of the *q* disks to one of the actual physical disks *p*. Thus, when DASDs are involved, the CPU interacts with the DASD on a logical disk level basis; SCSI protocol, for example, uses a logical disk, a logical unit number (LUN), and a logical block address (LBA).

Typically, computing systems include special-purpose system software, executed by the host CPU, to copy information stored on a DASD to a backup device, such as the DLT 2000, sold by Quantum Corp., which holds magnetic tape. Other back-up devices, including devices that use other types of "back-up media," may also be used. The copy residing on the back-up media is called a "back-up copy," and the process of copying the data to the tape or other media is called "back-up process." The back-up process reads data from the DASD into the main memory of the host CPU and then copies that data from the host CPU's main memory to the back-up media. Because the CPU software operates at a file-level of granularity, the conventional back-up facilities operate on a file level granularity, as well. For example, the back-up process may include instructions to the effect "back-up file *x*."

Back-up copies are useful if a DASD experiences a mechanical failure, software executing on the host CPU experiences a programming error, and the like. In such cases, the back-up copy of the data may be used to restore the computing system to a useful state that existed before the failure occurred. The process of restoring the DASD information to a useful state is called a "restore process."

Typically, other special-purpose software, executed on the host CPU, performs the restore process. The restore process reads data from the back-up media into the main memory of the host CPU and then copies that data from the main memory to the DASD. Like the back-up process, conventional restore facilities operate at a file-level of granularity. Thus, a conventional restore process may include an instruction to the effect "restore file *x* from the back-up copy."

Conventional back-up processes prohibit any updates to be performed to a file, while the back-up process is backing up that file. Because backing-up a file to back-up media may take a considerable time, this is a major inconvenience, because the DASD

is unavailable for normal, useful CPU activity during this time.

Some systems, such as the "Oracle Parallel Backup & Restore" system, sold by Oracle Corp. use a technique called "save while active" to improve the availability of the DASD during back-up. The host reads files from the DASD and writes them to the back-up device. If the host issues a write as part of its "normal" CPU activity, i.e., non-back-up activity, the write will update the DASD state, regardless of the back-up status. Any updates to the DASD state are logged in a temporary storage area. At the end of the back-up of the DASD data, the log file is also stored. Because of the intervening writes, the back-up image is non-coherent; i.e., the image does not correlate to the instant that back-up was started. When a restore is done, the DASD is restored with the DASD data. The log file is then searched, and any updates that affect a part of the file that was already backed up at the time of the update is then applied. Thus, the restored image is made more current. Consequently, a restore does not correlate to the instant that back-up was started, but rather includes all state changes that occurred during that window of time when back-up was being performed.

Summary

The invention provides a highly available DASD that can perform normal CPU activity while concurrently constructing a back-up copy of system state. Although certain advantages are attained by having the invention implemented in the DASD itself, advantages are also attained if the logic is implemented in CPU software. In short, either implementation attains the advantage of high-availability of the DASD.

When implemented in the DASD controller, the invention further realizes an advantage of off-loading back-up responsibilities from the CPU, thus improving overall system performance. This arrangement includes DASD logic to operate as an intelligent bus host on a bus that interconnects the CPU, the DASD, and a back-up device. That is, while conventional DASD arrangements merely respond to CPU commands, in this inventive arrangement, the DASD not only responds to CPU commands but initiates commands to an associated back-up device.

The inventive back-up logic normally copies data from the DASD to the back-up device in a predetermined sequence. If a write is received from the CPU during the back-up of DASD data, the back-up logic stores the DASD data at the DASD location targeted by the CPU write, out of sequence, to the back-up device. The CPU write is then handled, thus providing a high degree of availability.

The invention provides a coherent back-up image in which the DASD data that is backed up all corresponds to the same time instant in the computing system, in effect, providing a snapshot copy. The backed up DASD data may come from several files and disks, and in fact may come from several DASDs. The invention synchronizes the back-up to correspond to the same time instant. In addition, the back-up copy not only includes DASD data, but also includes title information, configuration information, and DASD firmware, all of which provide new advantages to system administrators and the like.

Brief Description of the Drawing

In the Drawing,

Fig. 1 an exemplary arrangement of the computer system in which a preferred embodiment of the invention may be realized;

Figs. 2A-B are a flow chart of the back-up logic of a preferred embodiment;

Fig. 3 is a flow chart of fast right buffer logic of a preferred embodiment;

Fig. 4 is a flow chart of destage logic of a preferred embodiment;

Figs. 5A-C are a flow chart illustrating the logic of preferred embodiment for managing the overall back-up process; and

Figs. 6A-B are a flow chart illustrating channel connection logic of a preferred embodiment.

Detailed Description

As will be more fully explained below, the instant invention provides unique logic to allow a back-up of a DASD(s) to be performed while concurrently allowing a host CPU to use the DASD(s). The unique logic achieves a coherent back-up image that includes DASD data and other useful information, correlated to a given synchronized instant when the back-up started.

Several arrangements may be used to realize the invention, as will be evident to skilled artisans upon reading the following. Figure 1 is but one exemplary arrangement in which the invention may be realized.

Computer system 100 includes host CPU 10, a DASD 15, and a back-up device 20. A bus 25, such as a SCSI bus, interconnects the above, all on the same bus channel. The system may also include any number of other arrangements of DASDs and back-up device pairs, e.g., 30 and 35, connected on their respective buses.

A control device 40 is used for maintenance and other system management functions, such as initiating the back-up or restore processes, discussed below. In a preferred embodiment, the control device 40 includes a personal computer with special management software, described below. Control device 40 communicates with DASD 15 via control bus 45, such as an RS 232 serial bus. If other pairs 30 and 35 are present, the control device 40 likewise communicates with the DASDs, for example, via buses 46-47.

Before beginning a back-up process, all the participating back-up devices 20 are loaded with an appropriate back-up media, and control device 40 loads all of the participating DASDs with user-supplied label data and DASD configuration data. The configuration information, among other things, describes the present arrangement of the DASD. For example, the DASD may be configured as a RAID5, RAID3, or RAID0 arrangement, all known in the art. The information provided will also described which logical disks are to be backed-up. Thus, unlike the conventional procedure, back-up is at a disk-level, not a file-level, of granularity. In addition, not all logical disks need be included in the back-up set. This process is sometimes referred to as "arming" the devices and may be performed shortly before or considerably before the actual back-up

occurs.

After the devices are armed, the control device 40 may issue a "begin backup command" to the various DASDs, e.g., 15, via the control buses 45-47. Among other things, the control device 40 may issue the "begin backup command" in response to manually entered commands, an internal timer residing in control device 40, or in some arrangements, host 10 (more below).

In response to receiving such a command, each DASD 15 suspends starting any "new" write commands. Any write commands that have been received and partially processed to thereby alter the state of the DASD and any writes that have been received by the DASD and acknowledged to the host are completed. These must be done because either the DASD or the CPU has altered its state. By completing the transaction the back-up image will be consistent with the state as CPU perceives it to be. These are called "partially completed writes." The CPU will receive acknowledgment of the partially completed writes once they are completed. The new writes will not be processed until later, once the back-up has begun (more below).

In addition, the DASD will initialize a bitmap data structure, discussed below, to indicate that all blocks to be backed up have not yet been backed up. Once the DASD performs the above, the DASD 15 sends a signal back to the control device 40, indicating that the DASD is "ready to proceed."

After receiving the "ready to proceed signal" from all participating DASDs, the control device 40 sends a proceed signal to all participating DASDs. In response to receiving the proceed command, each DASD 15 writes the label data and a time and date stamp to the back-up media. This is done so that the backed-up files may be easily located if a restore is later needed and so that the time of back-up is known.

At this point, the DASDs are placed in a state, which allows the DASDs to receive and process host CPU commands and concurrently perform the back-up process. If any new writes were suspended, as discussed above, these writes will also be processed now, as if they were just received from the CPU. This useful feature is a consequence of the inventive back-up logic, write-processing logic, and channel

connection logic discussed below.

The back-up image that is created will include a snapshot of the DASD data as it existed at the instant that writes were suspended. Thus, the back-up image will be consistent with the computing state at the synchronization instant. As will be explained below, the DASD can concurrently operate on all types of host commands, while the back-up is being performed. Thus, a coherent back-up image is achieved while also keeping the DASD(s) available to perform normal CPU activity.

Skilled artisans will appreciate that the following discussion is with particular reference to a firmware-implementation, of a DASD controller but that the invention is not limited to this manner of realization. Among other things, the invention may be realized completely in hardware or in hardware/ firmware combinations that will be apparent to skilled artisans upon reading the following.

Skilled artisans will also appreciate that modern controller designs are "multi-tasking." In short, multi-tasking involves a scheduler task that selects one of several firmware streams to use the controller's microprocessor. Many scheduling techniques are known, such as time-slicing and the like.

In the following description, the flow charts use a double slash, "//", e.g., item 205 of Fig. 2A, to identify predefined breakpoints in a logic stream. At these points, the firmware stream, then executing, returns control to the scheduler task. Depending on the state of the controller, the scheduler will then select an appropriate firmware stream for execution. The appropriate stream may be the stream previously executing or another. The following paragraphs describe several streams that may be multi-tasked, such as back-up stream, and a write stream. Other streams, such as a read stream, may also be multi-tasked.

The back-up logic is discussed with reference to Figures 2A-B. The process starts at step 200 and proceeds to step 210. In step 210, the back-up logic reads the lowest addressed block of the lowest ordered logical disk that is to be included in the backup from the corresponding DASD storage device into a predefined area of the DASD's controller memory. The information identifying the lowest ordered disk results

from the DASD configuration and other information previously entered from the control device 40, for example.

In step 220, back-up logic marks an entry in a data structure, stored in the controller memory, indicating that the block has been backed-up. In step 230, the back-up logic writes the stored block and its DASD address to the back-up media in back-up device 20. A preferred embodiment uses a bitmap data structure. By marking the bitmap first, the DASD becomes available sooner with respect to the block being backed up. However, the steps could be reversed. In this reversed arrangement, the DASD would wait for successful status from the back-up device before marking the bitmap. In this fashion, if any errors occurred during the write to the back-up device the bitmap would provide a useful picture of which data blocks were successfully backed-up.

In step 240, the logic determines whether the block just written was the highest addressed block of the highest ordered logical disk that is to be included in the backup. Similarly to the above, this information results from the previously entered configuration and other information entered from control device 40. If it is the highest block, the logic ends at step 290. If it is not, the logic proceeds to step 245.

In step 245, the backup logic determines whether the DASD 15 has any write requests that target data blocks that have not yet been backed up. These may include write requests that were suspended from being started during the startup of the back-up process, as discussed above, or they may be new writes, resulting from normal CPU activity. The mechanisms for determining this are disclosed below (*see infra* discussion of step 490 of Figure 4.). Step 250 branches on the determination.

If there are no such requests, the logic branches to step 255, which checks the bitmap to determine whether the next sequential block has already been backed up. For example, the block may have been already backed up as a result of a prior out of sequence write request. Step 260 branches on the result.

If the data for the next sequential block has already been backed up, the logic proceeds to step 270, which checks whether that was the highest addressed block. If it

is, the logic proceeds to step 290, which ends the back-up. If it is not, the logic proceeds back to steps 255 and 260 to check the next sequential block, as discussed above.

If the next sequential block has not been backed up, the logic proceeds to step 265, in which the logic reads that block from the DASD storage device into the controller memory. The logic then proceeds to steps 220-240, discussed above, which write the block and its DASD address to the back-up media.

If step 250 determines that a write has been received that has not yet been backed up, the logic branches to step 275. In step 275, the logic reads the block, targeted by the request, from the DASD storage device into controller memory. In step 280, the bitmap is marked to indicate that the block has been backed up. Then, in step 285, that block is written from controller memory to the back-up media along with its DASD address. The back-up logic then proceeds to step 245 to repeat the logic discussed above.

The read logic implemented by the controller is conventional. Reads may be processed by the controller firmware while backup is in progress without fear of compromising the data coherency of the back-up copy. Of course, known rules of read-write ordering must not be violated.

The write logic implemented by the controller includes conventional aspects for mapping a host-provided address, for example, of a write command to a particular location on a particular storage device, controlled by the DASD. Likewise, the controller uses various known techniques for implementing correcting codes, error recovery, and the like.

Some aspects of the write logic, however, are unique, particularly those aspects that relate to the back-up logic discussed above. The instant invention supports DASDs having a "fast write buffer" architecture, as well as architectures that lack fast write buffers.

Briefly, "fast write buffers" architectures include logic to receive write commands and their data from a host into non-volatile RAM on the DASD controller board. Once

the write and data are received, the DASD acknowledges the host, thereby freeing the host to perform other tasks and thereby improving the overall system performance.

The fast write buffer logic of a preferred embodiment is discussed with reference to Figure 3. The logic starts with step 300 and immediately proceeds to step 310. In step 310, the write command is accepted from the host CPU 10. Then, in step 320, the logic checks whether there is enough available space to store the write data in the write buffers. The logic branches on this determination in step 330. If there is insufficient space, the logic branches back to step 320. As is shown, this branching provides a predetermined breakpoint 335 at which other firmware streams may be scheduled. If sufficient space is available, the logic branches to step 340.

In step 340, the data is accepted from the host CPU 10 and stored into the write buffers. This includes sending an appropriate acknowledgment to the host CPU 10. The logic then ends in step 350.

Once data is in a write buffer, the write commands and data are destaged by destage logic to update the location of the storage device that maps to the target address. When the destage operation is completed the write buffer is then freed so that it can accept new writes.

The destage logic is described with reference to Figure 4. The destage logic begins at step 400 and immediately proceeds to step 410. In step 410, the destage logic determines whether the write buffer is empty, i.e., whether any write commands need destaging. The logic branches on this result in step 410.

If the write buffer is empty, the logic branches back to step 410. If the write buffer contains data and an address from a command, the logic branches to step 430.

In step 430, the destage logic selects the corresponding data of the command that needs destaging. Then, in step 440, the logic determines whether the target address of the write command is to a DASD location that has already been backed-up. As discussed above, this is performed by checking a corresponding location in a bitmap data structure. The logic then branches on this result in step 450.

If the target location has been backed up, the logic branches to step 460, the

write command is destaged and the target area is updated with the selected data. Because this location has already been backed up, the destage write cannot affect the coherency. Thus, the back-up logic need not be notified of this write. After performing the write, the destage logic returns to step 410.

If the target location has not been backed up yet, the destage logic branches to step 470. The DASD cannot be immediately updated without compromising the coherency of the back-up copy. Thus, in step 470, the logic determines whether the back-up logic has already been notified of this write. The logic then branches on this result in step 480.

If the back-up logic has been notified, the destage logic returns to step 410. If it has not, the logic notifies the back-up logic of this write so that the back-up logic can back the block up, out of sequence, as described above. The back-up logic stream will gain control due to the multi-tasking scheduling control.

The overall logic for the back-up process is explained with reference to Figures 5A-C. The logic starts at step 500 and immediately proceeds to step 505. In step 505, the control device 40 receives information, provided by a user, such as a system administrator, that specifies the desired DASD devices for back-up, the logical devices on those DASDs, and sets up labels to name the backed-up image. In step 510, the system administrator or other user arms the DASD devices for back-up and causes control device 40 to transmit the label data. In step 515, host CPU 10 sends a command to one of the DASDs to start back-up. For example, a "reserved use" command code of the SCSI protocol may be used for this purpose. For example, the CPU may want to initiate back-up at predetermined times or in response to certain events, such as the completion of certain CPU programs or at certain checkpoints. In step 520, the DASD that received the CPU command, in turn, notifies the control device 40 via its control bus 45 to start back-up. Then, step 525, control device 40 puts all participating DASD devices into a "write holding state" by issuing a "begin back-up command" to the DASDs, discussed above.

Each DASD eventually sends a "ready to proceed" signal back to the control

device 40, indicating that it is ready to proceed with the back-up. Step 530 indicates the time period in which control device 40 waits for the DASD devices to provide this notification. Step 535 indicates that a DASD device has provided a ready to proceed notification, and step 540 determines whether all DASD devices have provided this notification. If not, the logic branches back to step 530. If all the DASD devices have provided notification, the logic branches to step 545. In step 545, the control device commands all the DASD devices, participating in the back-up, to start to back-up and provides them with a time and date stamp.

In step 550, each participating DASD write labels, configuration information, and a time and date stamp to the back-up media on the back-up device 20. As explained above, the configuration information will described the DASD configuration, e.g., RAID5.

In step 555, each DASD device performs back-up, as described above, to its corresponding back-up device, while concurrently servicing normal CPU activity, such as read and write commands from host CPU 10. After the data is backed up, each DASD device stores a complete image of the controller firmware to the back-up media. The backed-up image of controller firmware has a variety of uses, including providing a more complete state of the system at the time of back-up. This may be useful for bug-tracking purposes, or for "cloning the DASD." For example, as will be explained below, the restore process may restore the backed-up data to the original DASD or another capable DASD. The combination of configuration information and firmware provided from the back-up allows for a complete cloning to a different DASD to be performed automatically. The new DASD will be automatically configured to the same DASD arrangement and use the same DASD firmware that existed at the time of back-up, thus completely restoring the DASD state at the synchronized back-up instant.

Once this is completed, the DASD device notifies control device 40 that the back-up is completed, in step 565. In step 570, the control device 40 displays the back-up status to a monitor, indicating the status of the back-up and the total time. Similarly, if any problems are experienced during back-up, the DASD notifies control device 40,

which displays status information to the monitor. The process ends in step 580.

Skilled artisans will appreciate that, unlike conventional arrangements, the above arrangement involves DASD/host interaction intelligence. Conventionally, DASDs are passive devices that respond to host CPU commands. The above architecture, however, requires the DASD to sometimes act as a target for a host, i.e., when receiving read and write commands from host CPU 10, and at other times requires the DASD 15 to act as a host to another target, i.e., when sending back-up writes to an associated back-up device 20.

To perform this, DASD controller firmware implements channel connection logic. In short, the channel connection logic connects the DASD controller to the bus 25, as either a host or a target, depending upon the operation to be performed.

The channel connection logic is more particularly described with reference to Figures 6A-B. The channel connection logic begins at step 600 and immediately proceeds to step 605. In step 605, the channel connection logic determines if a device is making a connection on the channel, such as if host CPU 10 is making a connection on the channel to send a CPU command. The logic branches on this result in step 610.

If no device is making a connection on the channel, channel connection logic branches to step 615. In step 615, the logic determines whether any of the firmware streams is requesting the use of the channel, for example, to return read data to the CPU 10 or to write back-up data to the back-up device 20. In step 620, the logic branches on this determination.

If no firmware stream is requesting the use of the channel, the logic branches back to step 605. If one of the firmware streams is requesting the use of the channel, the logic branches to step 625.

In step 625, channel connection logic determines whether the back-up logic is requesting the use of the channel. If so, the channel connection logic attempts to connect to the back-up device 20 as a "host" on the bus 25. Then, in step 635, the logic branches on the status of the connection request.

If access as a host is denied, the logic returns to step 605. If it succeeds, the

logic branches to step 640, where the write to the back-up device is performed as described above, until the channel is freed again. After performing the back-up of that block, the logic returns to step 605.

If step 625 determines that the back-up logic is not requesting the use of the channel, the logic branches to step 645. In step 645, the channel connection logic attempts to connect to the bus as a "target," connecting the DASD to the "host" that previously issued a command. In step 650, the connection logic branches on the status.

If connection is unsuccessful, the logic branches back to step 605. If connection is successful, the logic branches to step 655. In step 655, the DASD performs the corresponding operation, such as returning read data, carrying out the process until the channel is freed again.

If in step 610 the connection logic determines that a device is making a connection request, the logic branches to step 660. In step 660, the logic determines whether the connection request is made by the back-up device 20.

If so, the logic branches to step 665, which accepts the connection request as a bus host. Then, in step 670, the logic keeps the DASD as a host until the channel is freed again. For example, the back-up device 20 may make a connection request after it has successfully stored data on the back-up media and is ready to accept a new block of back-up data and an address.

If in step 660 the connection logic determines that the connection request is being made by a device other than the back-up device, the connection logic branches to step 675. In step 675, the connection logic connects to the bus as a target. In step 680, the connection logic keeps the DASD device as a target until the operation is complete. For example, the host CPU 10 may be sending a read command, or a write command.

Restoring the DASD state is straightforward given the above description. If a restore is needed, the back-up image is read from the back-up media to a DASD. The DASD involved in the restore need not be the same DASD and, instead, may be any

capable DASD, i.e., one with sufficient DASD storage to store the back-up copy. The restore DASD may load the backed-up image of firmware into its controller, the DASD data, and then configure itself as specified by the back-up copy. Although the restore image may be controlled by switches, for example, "don't load firmware," a preferred embodiment performs the above automatically in response to receiving a restore command, e.g., from control device 40.

During the restoring of the DASD data, the restore system will use a mapping process that corresponds to the back-up logic. For example, the preferred embodiment loads DASD data and its address in the back-up image. The restore logic will use the address to map the restore DASD data to its correct spot. However, as is discussed below, other back-up logic may also be used. The corresponding restore logic will be appreciated by skilled artisans.

The restore logic may be implemented in the CPU or also within a DASD. In either case, "restore sets" may be determined by analyzing the label information and the time/date stamp included in the back-up copy. This analysis may be performed by a system administrator, but a preferred embodiment determines the restore set through software logic that may reside in a control device or in the CPU. By implementing it as software logic, rather than requiring human intervention, the restore is automated. Among other things, this is helpful for systems that employ geographically remote systems. Using the instant invention, the back-up copy may be used to clone the DASD at a completely different computing site.

The preferred embodiment, discussed above, is but one exemplary arrangement. Many alternative arrangements will be apparent to skilled artisans, given the above description. For example, a DASD and its corresponding back-up device need not be connected on the same bus 25 (see Fig. 1) and instead could be connected via a separate path. Moreover, the host computer could act as the central control device using the existing paths to each DASD, rather than using control device 40. Moreover, the correspondence between DASD and back-up device need not be one-to-one; for example, two or more DASDs could be associated with a given back-up device and

vice-versa.

Concerning the back-up logic of Figure 2, skilled artisans will appreciate that, although the discussion refers to a normal sequence from lowest to highest, this sequence is arbitrary. Any predefined sequence may be used without loss of generality. Moreover, the sequence need not be predefined. The system may include sequence altering logic. For example, in response to a write request, the system may not only backup the immediately targeted write request but to alter the sequence so that DASD locations in the vicinity of the write target are backed up sooner.

Moreover, back-up logic need not write the DASD address and the data for each block being backed-up. Instead, other methods may be employed, which would trade-off simplicity of design for savings in back-up media storage. For example, a backup media could be used to only save blocks which are backed-up out of sequence. This media would hold the out-of sequence data with its address. The "normal" looping of backup would then only save data and would be later supplemented by the out of sequence media. Alternatively, a directory may be kept that gets written to the backup media at the end of the back-up or piece by piece in between backup blocks.

Furthermore, each backup block could be made up of a few of the DASD device's blocks, that is, there need not be a one-to-one correspondence.

In addition, although a preferred embodiment is described in which the DASD realizes the back-up logic, skilled artisans will appreciate that the inventive logic may also be realized in the host CPU. Although this arrangement does not off-load back-up responsibilities from the CPU, high availability of the DASDs and coherent back-ups may be achieved. In this arrangement, the CPU will keep the bitmap and perform the reads from DASD and writes to the back-up device.

The write logic was discussed with reference to fast write buffer architectures. Skilled artisans will appreciate that the invention is equally applicable to other architectures as well, including architectures employing holding buffers and architectures lacking holding buffers. In any of the embodiments, the material aspect is that any writes must be analyzed to determine whether it is targeting an area that has

been backed up (e.g., bitmap lookup.) If the location has been backed up, the write is scheduled to update the corresponding DASD location. If the location has not been backed up, the write must be temporarily delayed until that targeted location has been backed up out of sequence.

In addition, skilled artisans will appreciate the general applicability of the invention to other bus protocols. For example, conventional write transactions involve a command and address cycle followed by a number of data cycles. Future arrangements may include protocols that in effect issue write warnings, giving devices advance notice of an upcoming write. Hardware may be able to exploit this information by allocating resources accordingly, for example. Moreover, some bus transactions may have a name that includes "read" bu which skilled artisans recognize as involving writes. For example, a "read lock" operation both reads data and potentially changes the state of a variable. Thus, when the description and the claims refer to "writes," "write requests," "write commands," "write transactions," and the like, a skilled artisan will understand this term in its broadest possible sense, including write-warning-like commands suggested above and any transactions that alter DASD state.

As outlined above, the write holding state may be triggered in any number of manners, including manually entering commands at the control device 40 timeouts, by internal timers within the control device 40, or by host initiation via special bus commands and the like.

Moreover, the described invention ensures a time-synchronized coherent back-up in which multiple logical disks and files may be backed-up to correlate to the same instant. These disks and files can span multiple DASDs and the instant invention still provides synchronization as the control device is not limited to one DASD. Thus, if two files *a* and *b* are interrelated, those files may be backed-up to correspond to the same synchronized instant. That is; from the perspective of the CPU, the back-up images for files *a* and *b* correspond to the same instant in time, in short, the instant when writes were suspended; no fuzzy back-ups are encountered in which file a corresponds to one instant and file *b* another. Consequently, the restore logic will have a DASD image that

is consistently correlated to a given instant.

Furthermore, the system administrator is capable of requesting back-ups at a disk-level, not a file-level of granularity, with the instant invention. Often, the system administrator prefers this granularity, for example, if the system has informed of a "disk" error.

While the invention has been described in terms of a preferred embodiment in a specific system environment, those skilled in the art recognize that the invention can be practiced, with modification, in other and different hardware and software environments within the spirit and scope of the appended claims.

CLAIMS:

1. In a computing system having a CPU, a back-up device, and a direct access storage device (DASD) for storing and retrieving data blocks at corresponding DASD locations, a backup system comprising:

A. first logic for sequencing through the DASD locations and retrieving the data block stored at each DASD location and storing the retrieved data block to the back-up device;

B. second logic for recording information indicating which data blocks have had requests to store the data block to the back-up device and for using the recorded information to detect that a write transaction corresponds to a DASD location that has not yet had its corresponding data block requested to be stored to the back-up device;

C. third logic, cooperating with the first and second logic, for stalling the sequencing of the first logic, storing to the back-up device the data block at the DASD location, corresponding to the write transaction, detected by the second logic, and freeing the first logic to resume sequencing.

2. In a computing system having a CPU, a back-up device, and a direct access storage device (DASD) for storing and retrieving data blocks at correpsonding DASD locations, a backup system comprising:

A. first logic, residing in the DASD, for sequencing through the DASD locations and retrieving the data block stored at each DASD location and storing the retrieved data block to the back-up device;

B. second logic, residing in the DASD, for detecting that the DASD has received a write transaction to a DASD location that has not yet had its corresponding data block requested to be stored to the back-up device and for storing the data block at the detected DASD location to the back-up device.

3. The back-up system of claim 2, wherein the CPU includes means for communicating a begin back-up command to the DASD to initiate the first logic.

4. The back-up system of claim 2 further including at least one other DASD and at least one other back-up device, and wherein each DASD uses a corresponding one of the back-up devices as an associated back-up device for storing the DASD data blocks.

5. The back-up system of claim 2 wherein the computing system includes a connection channel for connecting the DASD to the CPU and the back-up device.

6. In a computing system having a CPU, a back-up device, and a direct access storage device (DASD) for storing and retrieving data blocks at corresponding DASD locations, a backup system comprising:

A. first logic, residing in the DASD, for sequencing through the DASD locations and retrieving the data block stored at each DASD location and storing the retrieved data block and information identifying its DASD location to the back-up device;

B. second logic, residing in the DASD, for recording information indicating which data blocks have had requests to store the data block to the back-up device;

C. third logic, residing in the DASD, for using the recorded information to detect that a write transaction, which has been received by the DASD, corresponds to a DASD location that has not yet had its data block requested to be stored to the back-up device;

D. fourth logic, residing in the DASD, for stalling the sequencing of the first logic, for storing to the back-up device the data block at the DASD location detected by the third logic and for storing information identifying its DASD location, and for freeing the first logic to resume sequencing.

7. The back-up system of claim 6 further comprising fifth logic, cooperating with the third and fourth logic, for ensuring that the fourth logic stores the DASD data block at the detected location only once.

8. The back-up system of claim 6, wherein the CPU includes means for communicating a begin back-up command to the DASD to initiate the first logic.

9. The back-up system of claim 6 further including at least one other DASD and at least one other back-up device, and wherein each DASD uses a corresponding back-up device as an associated back-up device for storing the DASD data blocks.

10. The back-up system of claim 6 wherein the computing system includes a connection channel for connecting the DASD to the CPU and the back-up device.

11. The back-up system of claim 10 further including channel connection logic, responsive to the first and fourth logic, for connecting to the channel as a target to service transactions from the CPU and for connecting to the channel as a host to issue store requests to the back-up device.

12. In a computing system having a CPU, a back-up device, and a direct access storage device (DASD) for storing and retrieving data blocks at corresponding DASD locations, a method comprising the steps of:

A. the DASD sequencing through the DASD locations and retrieving the data block stored at each DASD location

B. the DASD storing the retrieved data block and information identifying its DASD location to the back-up device;

C. the DASD recording information indicating which data blocks have had requests to store the data block to the back-up device;

D. the DASD detecting that a write transaction has been received by the DASD to a DASD location that has not yet had its corresponding data block requested to be stored to the back-up device;

E. the DASD stalling the sequencing of step (A);

F. the DASD storing to the back-up device the data block at the DASD location detected by step (D) and storing information identifying its DASD location;

G. the DASD resuming the sequencing of step (A).

13. The method of claim 12 further comprising the step of

H. ensuring that step (F) stores the DASD data block at the detected location only once.

14. The method of claim 12 further comprising the step of

I. the CPU communicating a begin back-up command to the DASD to initiate the first logic.

15. The method of claim 12, wherein the computing system further includes at least one other DASD and at least one other back-up device, and wherein in step (B) one of the DASDs uses a corresponding one of the back-up devices as an associated back-up device for storing the DASD data blocks.

16. The method of claim 12 wherein the computing system includes a connection channel for connecting the DASD to the CPU and the back-up device, and wherein the method further comprises the steps of

J. the DASD connecting to the channel as a target to service transactions from the CPU; and

K. the DASD connecting to the channel as a host to issue store requests to the back-up device.

17. In a computing system having a CPU, a back-up device, and a direct access storage device (DASD) for storing and retrieving data blocks at corresponding DASD locations, wherein the CPU, the back-up device, and the DASD are interconnected by a bus, and wherein the DASD includes a controller board having a processor and a memory for holding firmware, executable by the processor, a backup system comprising:

A. first firmware logic for connecting the DASD to the bus as a host in response to host connection requests by other firmware logic and for connecting the DASD to the bus as a target in response to target connection requests by other firmware logic;

B. second firmware logic for sequencing through the DASD locations in a predefined sequence and for retrieving the data block stored at each DASD location and for storing the retrieved data block and its DASD location to the back-up device, the second firmware logic issuing a host connection request to the first logic so that the DASD connects to the bus as a host to provide the store request to the back-up device;

C. third firmware logic for managing a bitmap data structure in the controller memory, the bitmap having an entry for each DASD location, and third firmware logic marking a corresponding entry in the bitmap for each DASD block stored to the back-up device;

D. fourth firmware logic, cooperating with the third firmware logic, for detecting that a write transaction, received by the DASD, targets a DASD location that has not yet had its corresponding data block stored to the back-up device;

E. fifth firmware logic, cooperating with the fourth firmware logic, for stalling the sequencing of the first firmware logic, for storing to the back-up device the data block at the DASD location detected by the fourth firmware logic and for storing its DASD location, and for freeing the first firmware logic to resume sequencing.

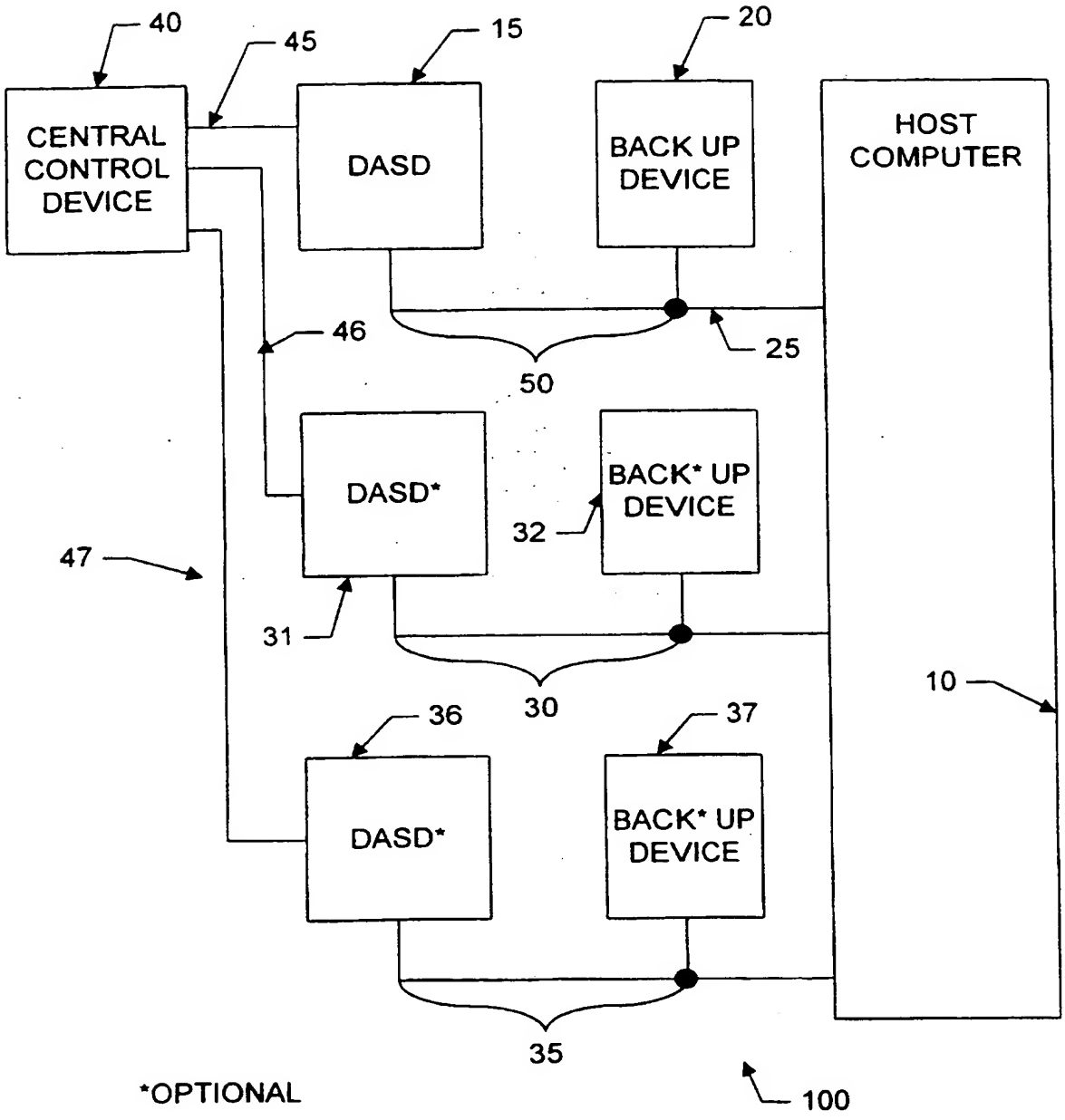

40
45
15
20
CENTRAL CONTROL DEVICE
DASD
BACK UP DEVICE
HOST COMPUTER
46
25
50
DASD*
BACK* UP DEVICE
32
47
31
30
10
36
37
DASD*
BACK* UP DEVICE
35
*OPTIONAL
100

FIG. 1

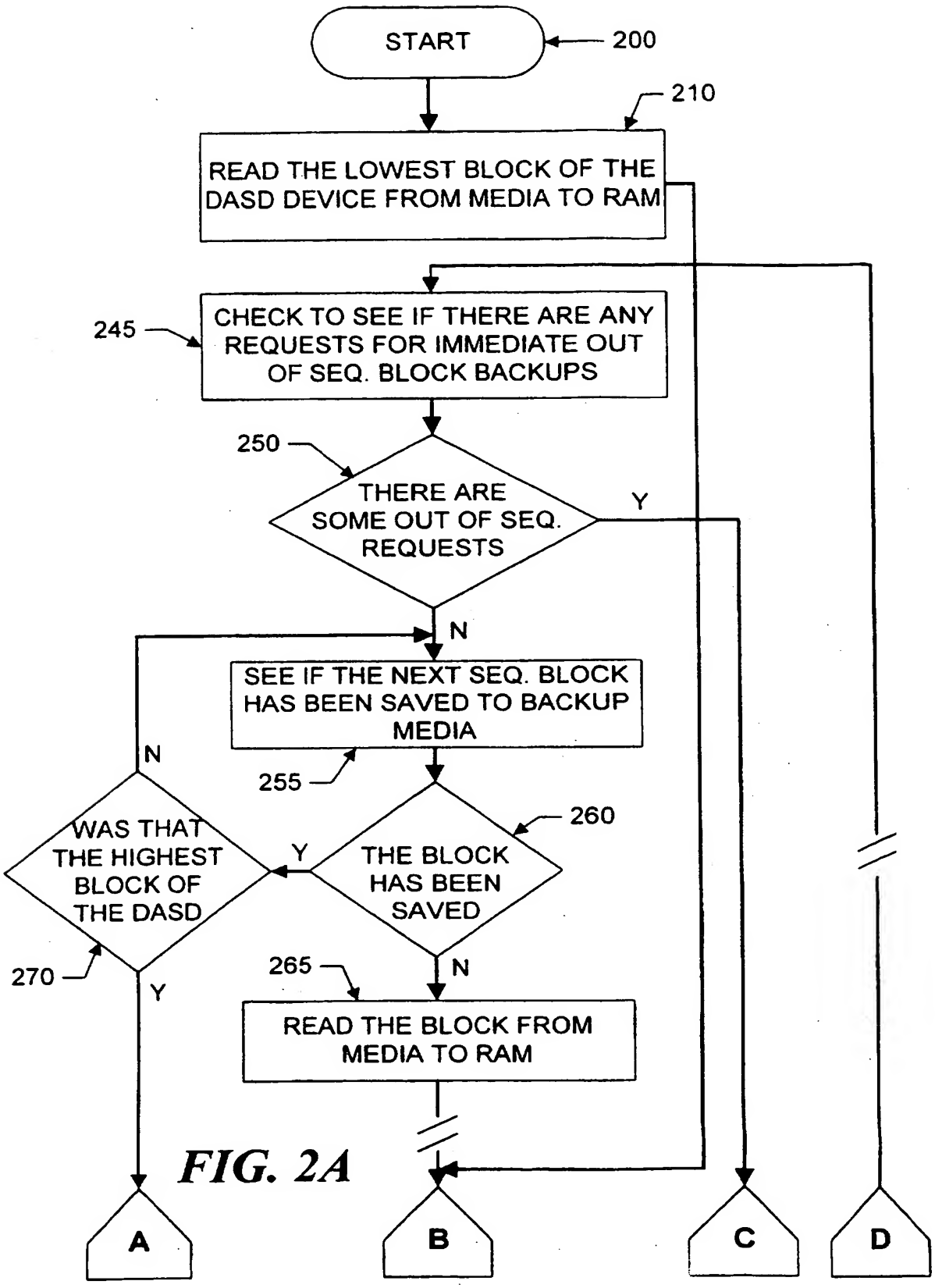

START
200
210
READ THE LOWEST BLOCK OF THE DASD DEVICE FROM MEDIA TO RAM
245
CHECK TO SEE IF THERE ARE ANY REQUESTS FOR IMMEDIATE OUT OF SEQ. BLOCK BACKUPS
250
THERE ARE SOME OUT OF SEQ. REQUESTS
Y
N
SEE IF THE NEXT SEQ. BLOCK HAS BEEN SAVED TO BACKUP MEDIA
255
260
THE BLOCK HAS BEEN SAVED
Y
WAS THAT THE HIGHEST BLOCK OF THE DASD
N
270
Y
265
N
READ THE BLOCK FROM MEDIA TO RAM
A
B
C
D

FIG. 2A

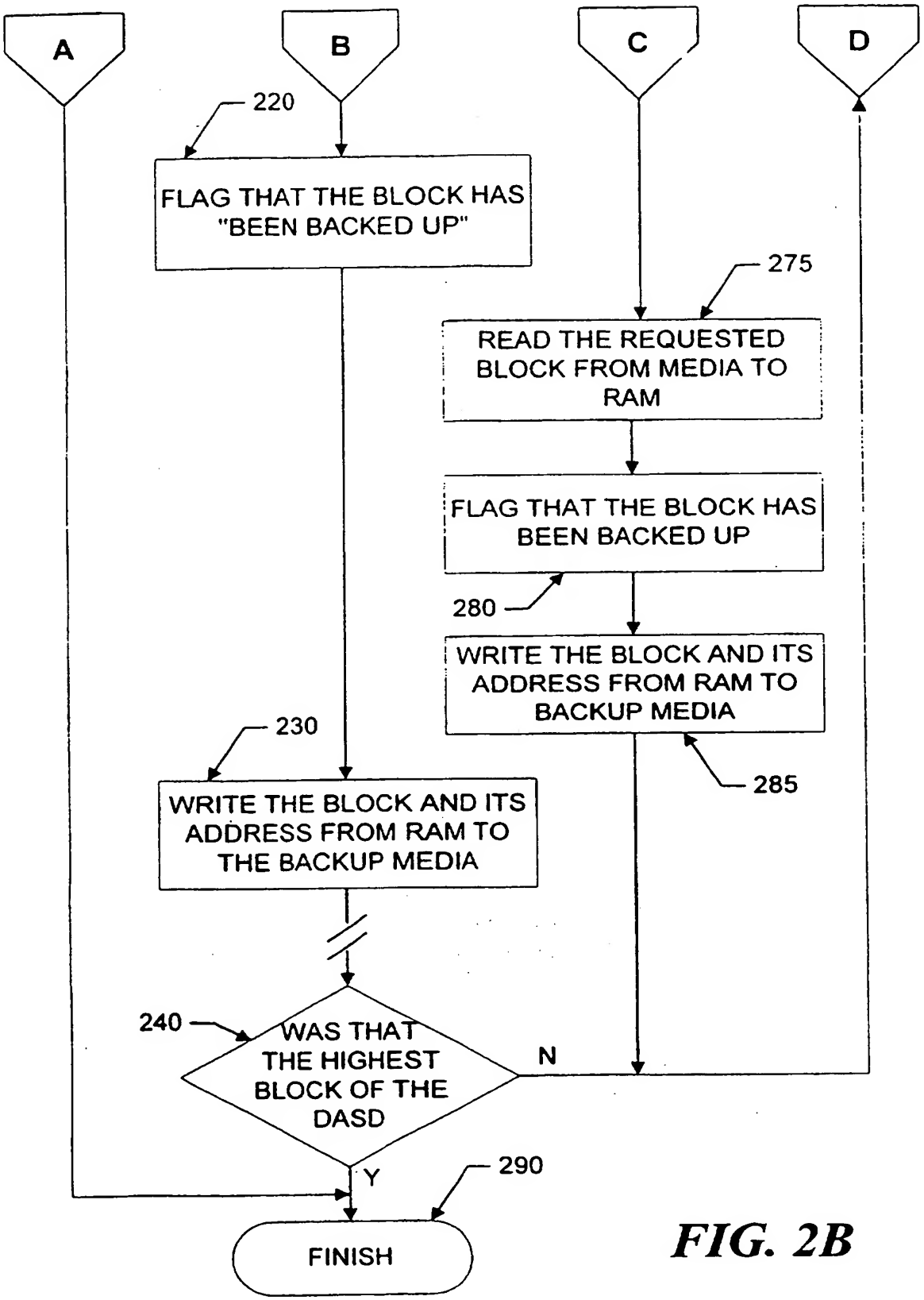

A
B
C
D
220
FLAG THAT THE BLOCK HAS "BEEN BACKED UP"
275
READ THE REQUESTED BLOCK FROM MEDIA TO RAM
FLAG THAT THE BLOCK HAS BEEN BACKED UP
280
WRITE THE BLOCK AND ITS ADDRESS FROM RAM TO BACKUP MEDIA
285
230
WRITE THE BLOCK AND ITS ADDRESS FROM RAM TO THE BACKUP MEDIA
240
WAS THAT THE HIGHEST BLOCK OF THE DASD
N
Y
290
FINISH

FIG. 2B

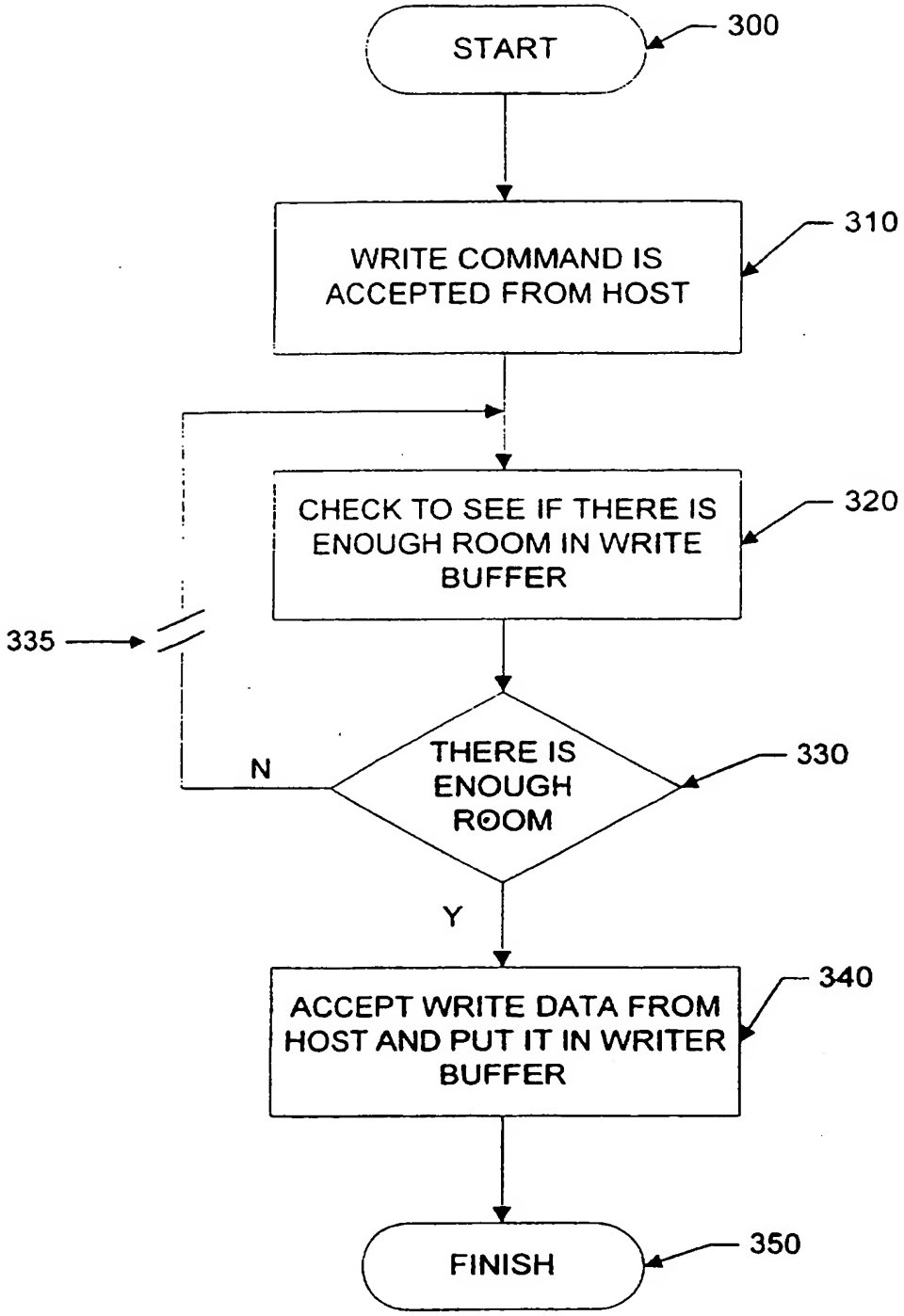

START
300
WRITE COMMAND IS ACCEPTED FROM HOST
310
CHECK TO SEE IF THERE IS ENOUGH ROOM IN WRITE BUFFER
320
335
THERE IS ENOUGH ROOM
330
N
Y
ACCEPT WRITE DATA FROM HOST AND PUT IT IN WRITER BUFFER
340
FINISH
350

FIG. 3

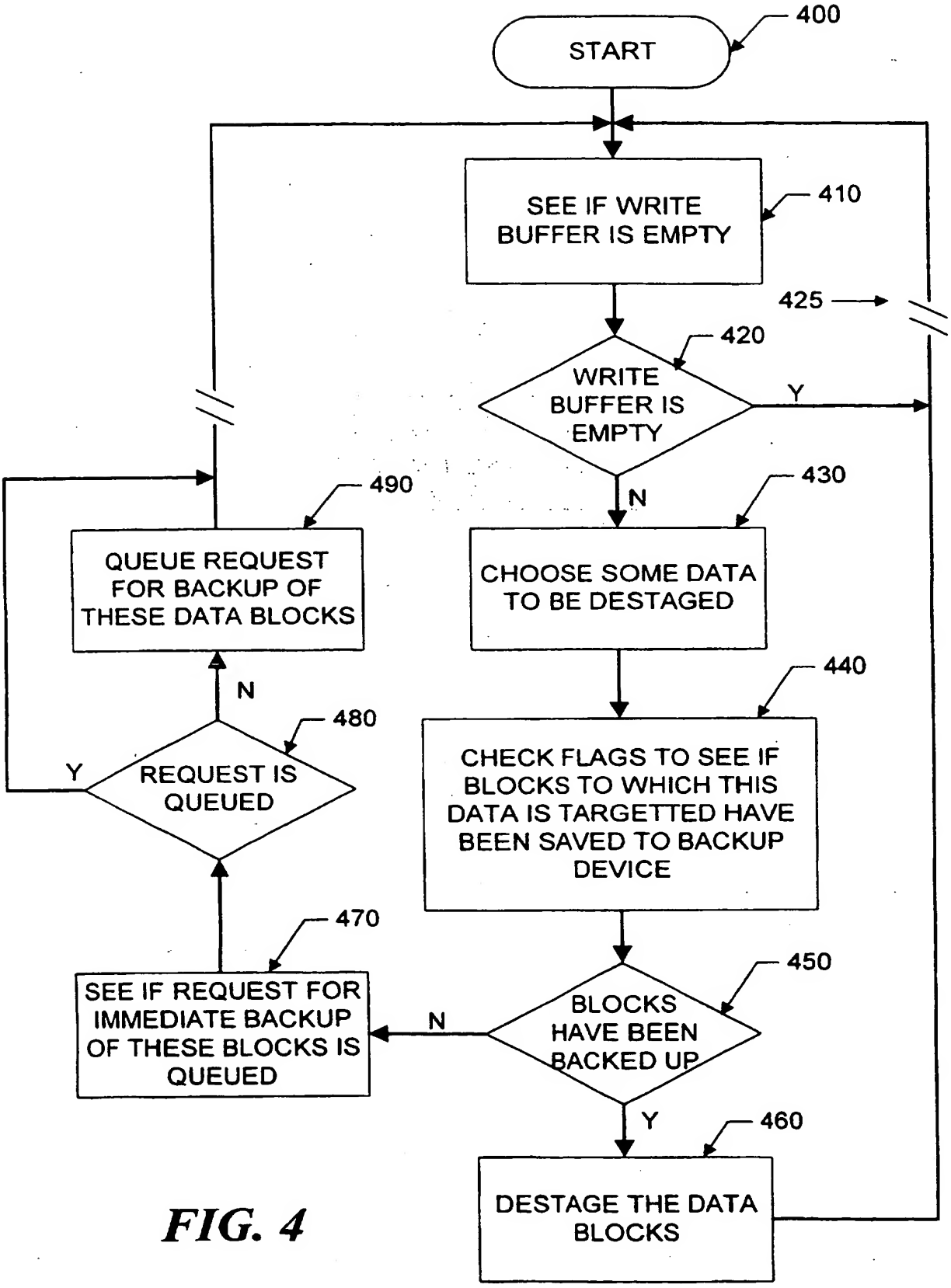

400
START
410
SEE IF WRITE BUFFER IS EMPTY
425
420
WRITE BUFFER IS EMPTY
Y
N
430
CHOOSE SOME DATA TO BE DESTAGED
440
CHECK FLAGS TO SEE IF BLOCKS TO WHICH THIS DATA IS TARGETTED HAVE BEEN SAVED TO BACKUP DEVICE
450
BLOCKS HAVE BEEN BACKED UP
N
Y
460
DESTAGE THE DATA BLOCKS
470
SEE IF REQUEST FOR IMMEDIATE BACKUP OF THESE BLOCKS IS QUEUED
480
REQUEST IS QUEUED
Y
N
490
QUEUE REQUEST FOR BACKUP OF THESE DATA BLOCKS

FIG. 4

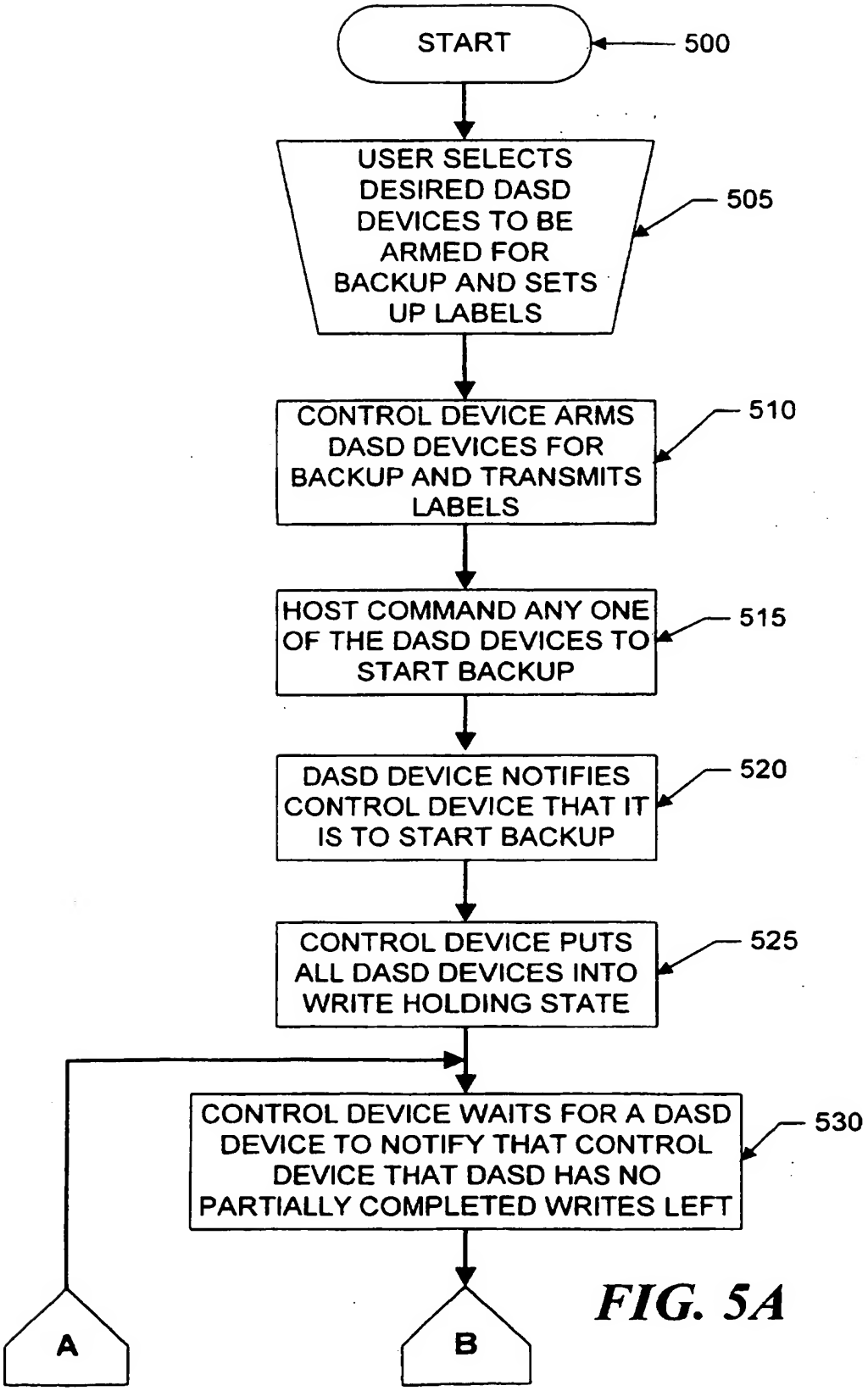

START
500
USER SELECTS DESIRED DASD DEVICES TO BE ARMED FOR BACKUP AND SETS UP LABELS
505
CONTROL DEVICE ARMS DASD DEVICES FOR BACKUP AND TRANSMITS LABELS
510
HOST COMMAND ANY ONE OF THE DASD DEVICES TO START BACKUP
515
DASD DEVICE NOTIFIES CONTROL DEVICE THAT IT IS TO START BACKUP
520
CONTROL DEVICE PUTS ALL DASD DEVICES INTO WRITE HOLDING STATE
525
CONTROL DEVICE WAITS FOR A DASD DEVICE TO NOTIFY THAT CONTROL DEVICE THAT DASD HAS NO PARTIALLY COMPLETED WRITES LEFT
530
A
B

FIG. 5A

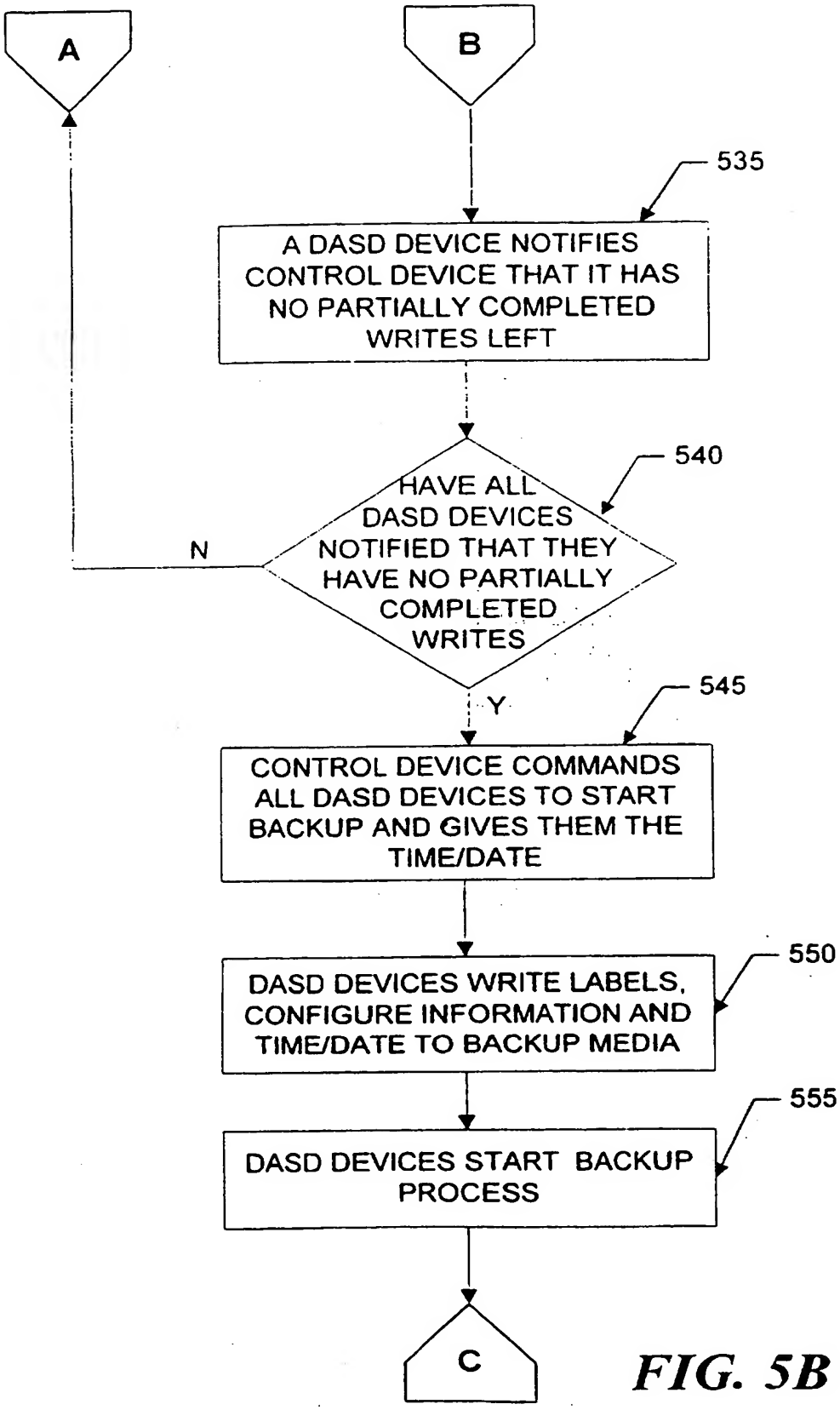

A
B
535
A DASD DEVICE NOTIFIES CONTROL DEVICE THAT IT HAS NO PARTIALLY COMPLETED WRITES LEFT
540
HAVE ALL DASD DEVICES NOTIFIED THAT THEY HAVE NO PARTIALLY COMPLETED WRITES
N
Y
545
CONTROL DEVICE COMMANDS ALL DASD DEVICES TO START BACKUP AND GIVES THEM THE TIME/DATE
550
DASD DEVICES WRITE LABELS, CONFIGURE INFORMATION AND TIME/DATE TO BACKUP MEDIA
555
DASD DEVICES START BACKUP PROCESS
C

FIG. 5B

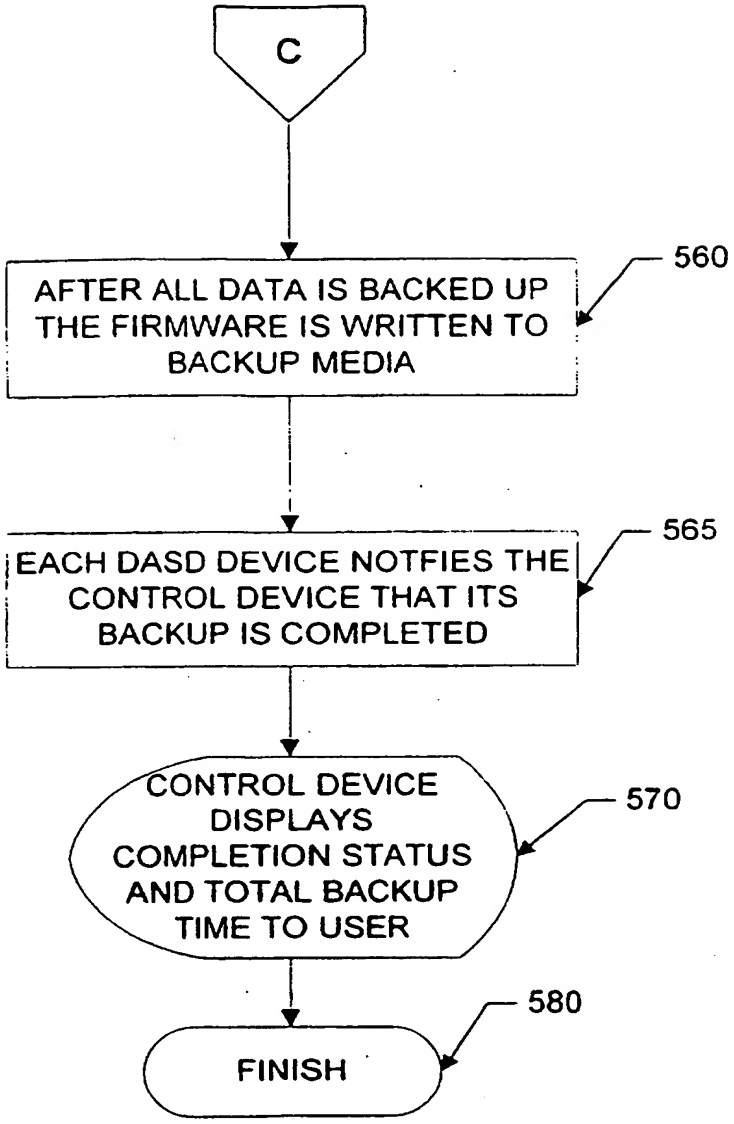

C
AFTER ALL DATA IS BACKED UP THE FIRMWARE IS WRITTEN TO BACKUP MEDIA
560
EACH DASD DEVICE NOTFIES THE CONTROL DEVICE THAT ITS BACKUP IS COMPLETED
565
CONTROL DEVICE DISPLAYS COMPLETION STATUS AND TOTAL BACKUP TIME TO USER
570
FINISH
580

FIG. 5C

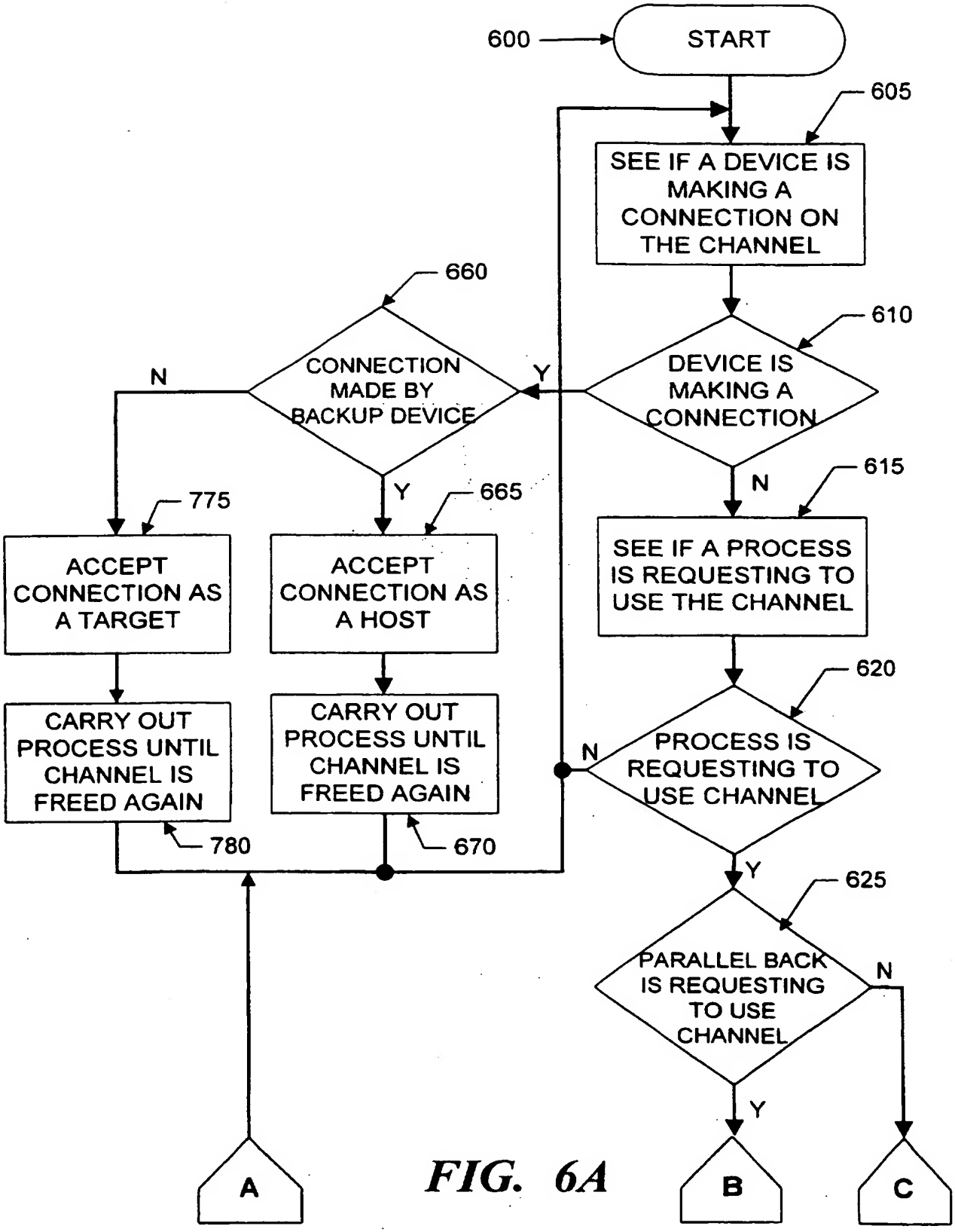

600
START
605
SEE IF A DEVICE IS MAKING A CONNECTION ON THE CHANNEL
610
DEVICE IS MAKING A CONNECTION
Y
N
660
CONNECTION MADE BY BACKUP DEVICE
N
Y
775
ACCEPT CONNECTION AS A TARGET
665
ACCEPT CONNECTION AS A HOST
615
SEE IF A PROCESS IS REQUESTING TO USE THE CHANNEL
620
PROCESS IS REQUESTING TO USE CHANNEL
N
Y
CARRY OUT PROCESS UNTIL CHANNEL IS FREED AGAIN
780
CARRY OUT PROCESS UNTIL CHANNEL IS FREED AGAIN
670
625
PARALLEL BACK IS REQUESTING TO USE CHANNEL
N
Y
A
B
C

FIG. 6A

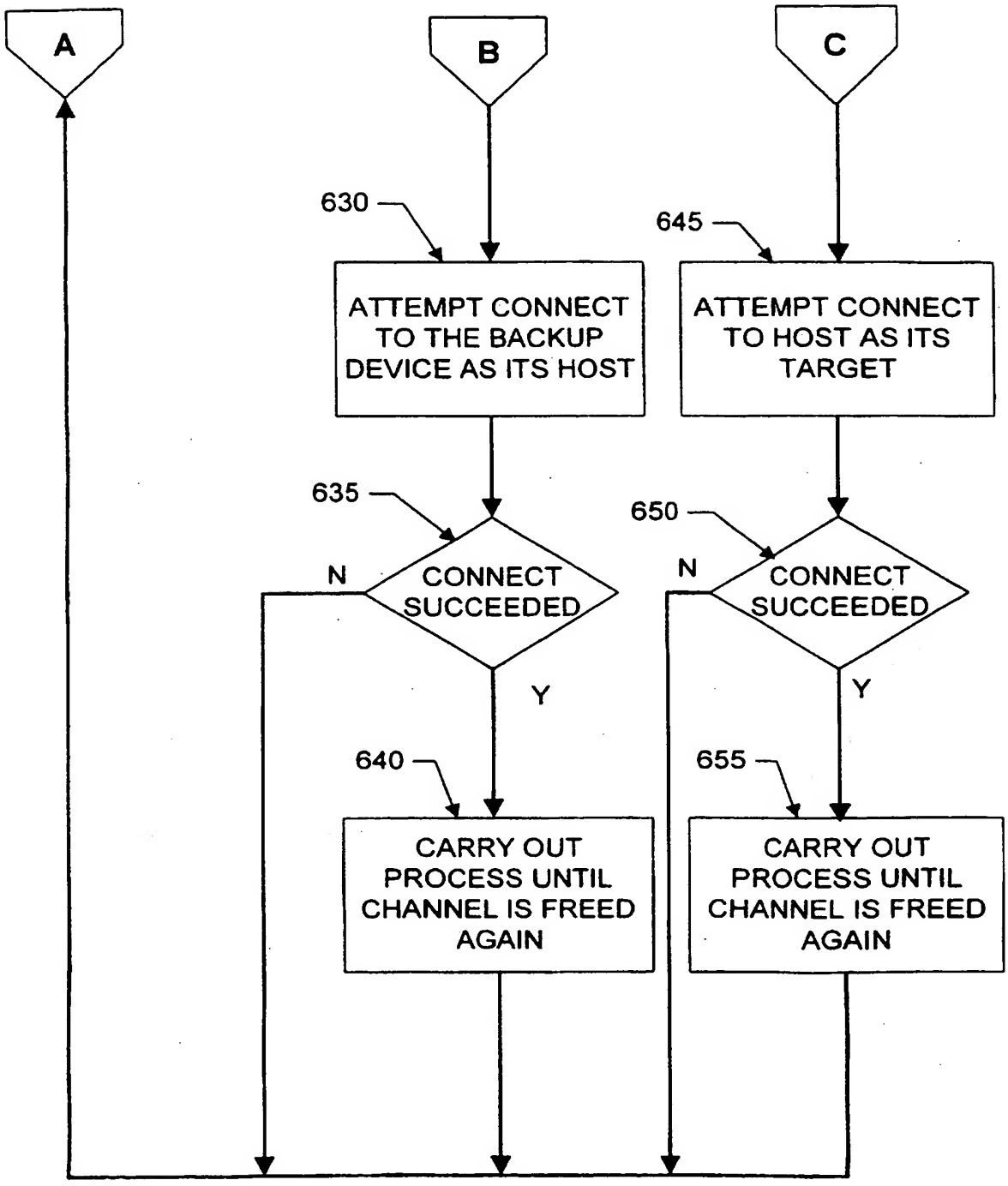

A
B
C
630
645
ATTEMPT CONNECT TO THE BACKUP DEVICE AS ITS HOST
ATTEMPT CONNECT TO HOST AS ITS TARGET
635
650
N
CONNECT SUCCEEDED
N
CONNECT SUCCEEDED
Y
Y
640
655
CARRY OUT PROCESS UNTIL CHANNEL IS FREED AGAIN
CARRY OUT PROCESS UNTIL CHANNEL IS FREED AGAIN

FIG. 6B

INTERNATIONAL SEARCH REPORT

International Application No PCT/US 96/19927

A. CLASSIFICATION OF SUBJECT MATTER
IPC 6 G06F11/14

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 6 G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	IBM TECHNICAL DISCLOSURE BULLETIN, vol. 37, no. 4B, 1 April 1994, pages 145-147, XP000451203 "CONCURRENT COPY" see the whole document ---	1-3,6-8, 12-14,17
A	WO 93 08529 A (IBM DEUTSCHLAND ;IBM (DE)) 29 April 1993 see the whole document ---	1-3,6-8, 12-14,17
A	US 5 163 148 A (WALLS KEITH) 10 November 1992 see the whole document ---	1,2,6, 12,17
A	EP 0 332 210 A (HITACHI LTD) 13 September 1989 see the whole document ---	1,2,6, 12,17

-/--

[X] Further documents are listed in the continuation of box C.

[X] Patent family members are listed in annex.

* Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

1

Date of the actual completion of the international search	Date of mailing of the international search report
17 April 1997	06. 05. 97

Name and mailing address of the ISA	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax (+31-70) 340-3016	Herreman, G

Form PCT/ISA/210 (second sheet) (July 1992)

INTERNATIONAL SEARCH REPORT

Interr nal Application No
PCT/US 96/19927

C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	US 5 212 784 A (SPARKS CLYDE R) 18 May 1993 see abstract -----	5,10,11, 16,17

INTERNATIONAL SEARCH REPORT

.formation on patent family members

Interr nal Application No
PCT/US 96/19927

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
WO 9308529 A	29-04-93	CA 2071346 A	19-04-93
		CN 1025381 B	06-07-94
		EP 0608255 A	03-08-94
		JP 5210555 A	20-08-93
		KR 9514175 B	22-11-95
US 5163148 A	10-11-92	NONE	
EP 0332210 A	13-09-89	JP 1231150 A	14-09-89
		JP 7043676 B	15-05-95
		DE 68925653 D	28-03-96
		DE 68925653 T	29-08-96
		US 5226157 A	06-07-93
US 5212784 A	18-05-93	NONE	